IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of: **Edward Triou, Jr., André Milbradt, Osarumwemse U. Agbonile, and Affan Ashad Dar**

Confirmation No.: **7147**

Serial No.: **10/828,947**

Group Art Unit: **2114**

Filing Date: **April 21, 2004**

Examiner: **Paul F. Contino**

For: **SYSTEMS AND METHODS FOR AUTOMATED CLASSIFICATION AND ANALYSIS OF LARGE VOLUMES OF TEST RESULT DATA**

Mail Stop Appeal-Brief Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

APPELLANT'S BRIEF PURSUANT TO 37 C.F.R. § 41.37

This brief is being filed in support of Appellant's appeal from the rejections of claims 1-12, 14, 22-24, and 27 in a final action dated April 19, 2007. A Notice of Appeal was filed on October 18, 2007.

1. REAL PARTY IN INTEREST

MICROSOFT CORPORATION is the real party in interest in the present application. The inventors in the present application assigned their interest to MICROSOFT CORPORATION as recorded by the United States Patent and Trademark Office ("USPTO") on April 21, 2004 at reel 015255, frame 0584.

2. RELATED APPEALS AND INTERFERENCES

There are no related appeals or interferences.

3. STATUS OF CLAIMS

Claims 1-12, 14, 22-24, and 27 are rejected. The rejections of claims 1-12, 14, 22-24, and 27 are appealed.

4. STATUS OF AMENDMENTS

A response was filed on June 19, 2007, subsequent to the close of prosecution. In the response, Applicants proposed amendments to independent claims 1 and 7. These amendments were not entered, as stated in the Advisory Action of June 28, 2007.

5. SUMMARY OF CLAIMED SUBJECT MATTER

Today's software testing produces an enormous number of test results, because many software operations are tested in many test scenarios. For example, a software operation may be tested against every operating system environment in which the software may be deployed. As would be expected, many tests produce many test results. Test results can be unwieldy due to the sheer volume. Embodiments of the invention thus provide for automated classification and analysis of large volumes of test result data.

One aspect of the independent claims is that test result data is classified around representative test failures. This notion is present in each of the independent claims as will be apparent from a review of the claims. Independent claims 1 and 7, for example, provide for "linking said list of operating systems to said representative test failure in said database." Independent claim 8 provides "linking an operating system identification from said test result file to said failure characteristics if said data from a test result file matches said failure characteristics." Independent claim 22 provides "adding said operating system identifier to a list of operating system identifiers associated with said single failure."

The notion of classifying test result data around representative test failures is supported in Applicants specification for example at Fig. 3 and corresponding text at paragraphs 0025 and 0045-0053. Specific support for linking an operating system identification to a representative test failure can be found for example at Fig. 0012 and corresponding paragraphs 0054-0059.

The independent claims also recite a variety of elements in addition to those discussed above. These elements are supported in Applicants' specification at least as follows:

Independent claims 1 and 7

Independent claims 1 and 7 contain substantially similar elements – the only difference is that claim 1 is directed to a method while claim 7 is directed to a computer

readable medium. Therefore, the following information provided with reference to claim 1 shall also suffice for claim 7.

1. A method for analyzing test results, comprising:

reading test result data corresponding to at least two test failures; See Fig. 12 (Find matching failure), Paragraph 0040 and 0041, and also Fig. 3 and corresponding text at paragraphs 0025 and 0045-0053 (Collapsing Failures).

wherein a test failure comprises a failed attempt by a software application to conduct an electronic operation on a computer equipped with an operating system; See paragraph 0041 (A first line in the input result file 200 identifies a test that was conducted, "open a file."), and also Fig. 3, illustrating configurations with various operating systems.

wherein said test result data identifies an operating system associated with each test failure; See Fig. 0012 and corresponding paragraphs 0054-0059. See also failure configurations 401 in Fig. 4 and corresponding list 408 that identifies operating systems (XP, NT5, ME)

determining a representative test failure in said test result data, said representative test failure corresponding to a first failed operation; See Fig. 3 and corresponding text at paragraphs 0025 and 0045-0053.

determining at least one related test failure corresponding to a second failed operation, wherein said second failed operation is a same operation as said first failed operation; See Fig. 3 and corresponding text at paragraphs 0025 and 0045-0053.

parsing said test result data to generate a list of operating systems corresponding to said representative test failure and said at least one related test failure; and Paragraph 0040 and 0041, See also failure configurations 401 in Fig. 4 and corresponding list 408 that identifies operating systems (XP, NT5, ME).

linking said list of operating systems to said representative test failure in said database. See Fig. 0012 and corresponding paragraphs 0054-0059. See also failure configurations 401 in Fig. 4 and corresponding list 408 that identifies operating systems (XP, NT5, ME).

Independent claim 8

8. A method for classifying test results, comprising:

extracting data from a test result file, wherein said test result file identifies a failed

attempt by a software application to conduct an electronic operation on a computer equipped with an operating system; See failure configurations 401 in Fig. 4 and corresponding list 408 that identifies operating systems (XP, NT5, ME) , see also paragraph 0040 and 0041.

comparing said data to failure characteristics stored in a database; and See Fig. 12 (Search for matching failure characteristics across scenarios), Fig. 3 and corresponding text at paragraphs 0025 and 0045-0053.

linking an operating system identification from said test result file to said failure characteristics if said data from a test result file matches said failure characteristics. See Fig. 0012 and corresponding paragraphs 0054-0059. See also failure configurations 401 in Fig. 4 and corresponding list 408 that identifies operating systems (XP, NT5, ME).

Independent claim 22

22. A method for classifying test results, comprising:

extracting data from a test result file, wherein said test result file identifies a failed attempt by a software application to conduct an electronic operation on a computer equipped with an operating system; See failure configurations 401 in Fig. 4 and corresponding list 408 that identifies operating systems (XP, NT5, ME) , see also paragraph 0040 and 0041.

comparing said data from a test result file to failure characteristics stored in a database, wherein first data that identifies a test operation is used in said comparison and second data that identifies a test scenario comprising at least an operating system identifier is not used in said comparison; and See Fig. 12 (Search for matching failure characteristics across scenarios), Fig. 3 and corresponding text at paragraphs 0025 and 0045-0053.

if a match is discovered from said comparing, identifying said data from a test result file and said failure characteristics as a single failure in a Graphical User Interface (GUI), and adding said operating system identifier to a list of operating system identifiers associated with said single failure. See Failure Viewer Tree 1118 in the box labeled "GUI" in Fig. 11. See Fig. 0012 and corresponding paragraphs 0054-0059. See also failure configurations 401 in Fig. 4 and corresponding list 408 that identifies operating systems (XP, NT5, ME).

6. ISSUES TO BE REVIEWED ON APPEAL

Applicants appeal the rejection of claims 1-12, 14, 22-24, and 27 under 35 U.S.C. §§ 102 and 103 as allegedly either anticipated by U.S. Pub. 2002/0124213 (Ahrens), or

unpatentable over Ahrens in view of one or more of U.S. Pat. 6,438,716 (Snover), and 7,058,860 (Miller).

Specifically, Applicants appeal:

- The rejection of claims 1-3 and 7 as allegedly anticipated by Ahrens.
- The rejection of claims 4-6 as allegedly unpatentable over Ahrens in view of Snover.
- The rejection of claims 8-11 and 14 as allegedly unpatentable over Ahrens in view of Miller.
- The rejection of claims 12, 22-24, and 27 as allegedly unpatentable over Ahrens in view of Snover and Miller.

7. ARGUMENT

Contrary to the assertions in the Official Action of April 19, 2007 (See Official Action page 3, third paragraph; and Official Action page 6, paragraph beginning with "linking an operating system), Ahrens does not disclose linking operating system information to test failures as specified in Applicants' claims.

This element is provided in Applicants' independent claims as described in the above "summary of claimed subject matter" section, namely: Independent claims 1 and 7 recite "linking said *list of operating systems* to said representative test failure in said database," and also "parsing said test result data to generate a *list of operating systems* corresponding to said representative test failure and said at least one related test failure." Independent claim 8 recites "linking an operating system identification from said test result file to said failure characteristics *if said data from a test result file matches said failure characteristics*" (While Ahrens does not conditionally link the operating system identifier to other information, such as failure characteristics. Instead Ahrens maintains an error log that always has an operating system identifier field.) Independent claim 22 recites "adding said operating system identifier to a *list of operating system identifiers* associated with said single failure."

Ahrens provides an operating system identifier when reporting an error event, as disclosed in Ahrens paragraph 0049. If the error occurs again, Ahrens increments a counter, as disclosed in Ahrens 0051. However, besides the operating system that initially reports an error, Ahrens does not maintain information about which other operating systems experience the error. Ahrens 0051 states, "[i]f an event log entry has already been logged, a counter is

incremented to indicate the number of times an error log entry is received for this particular error event. If no error event log entry has already been logged, this event log entry will be logged."

This clear failing of the Ahrens reference is only the tip of the iceberg of the broader failing of Ahrens to teach or describe Applicants' claimed invention. Ahrens pertains to error reporting, and not to test result classification. Ahrens makes no reference to test result data or test failures, while Applicants' independent claims make numerous references to "test result data" and "test failures."

The Official Action does not allege that Miller, Snover, or the other references of record cure the above described deficiency of Ahrens, and indeed they do not. Therefore the Official Action has omitted one or more essential elements needed for a prima facie rejection. The various dependent claims define over Ahrens for the same reason.

Applicants respectfully request an appropriate order to withdraw the rejections.

Date: February 6, 2008

/Nathaniel Gilder/
Nathaniel Gilder
Registration No. 53,233

Woodcock Washburn LLP
Cira Centre
2929 Arch Street, 12th Floor
Philadelphia, PA 19104-2891
Telephone: (215) 568-3100
Facsimile: (215) 568-3439

CLAIMS APPENDIX

The following presents the claims to be reviewed on appeal without the amendments of June 19, 2007, as required under 37 CFR 41.67(c)(2):

1. A method for analyzing test results, comprising:

reading test result data corresponding to at least two test failures;

wherein a test failure comprises a failed attempt by a software application to conduct an electronic operation on a computer equipped with an operating system;

wherein said test result data identifies an operating system associated with each test failure;

determining a representative test failure in said test result data, said representative test failure corresponding to a first failed operation;

determining at least one related test failure corresponding to a second failed operation, wherein said second failed operation is a same operation as said first failed operation;

parsing said test result data to generate a list of operating systems corresponding to said representative test failure and said at least one related test failure; and

linking said list of operating systems to said representative test failure in said database.

2. A method according to claim 1, wherein at least a portion of said method is accomplished by a stored procedure in a database.

3. A method according to claim 1, wherein said test result data identifies a computer processor associated with each test failure, and further comprising including computer processor identification in said list of operating systems.

4. A method according to claim 1, further comprising exposing said at least one representative test failure through a Graphic User Interface ("GUI").

5. (Previously Presented) A method according to claim 4, further comprising marking said at least one representative test failure in said GUI as an expected failure.

6. (Previously Presented) A method according to claim 5, further comprising deemphasizing said at least one representative test failure in said GUI with respect to any unexpected failures.

7. A computer readable medium bearing instructions automated test result analysis, comprising:

instructions for reading test result data corresponding to at least two test failures;

wherein a test failure comprises a failed attempt by a software application to conduct an electronic operation on a computer equipped with an operating system;

wherein said test result data identifies an operating system associated with each test failure;

instructions for determining a representative test failure in said test result data, said representative test failure corresponding to a first failed operation;

instructions for determining at least one related test failure corresponding to a second failed operation, wherein said second failed operation is a same operation as said first failed operation;

instructions for parsing said test result data to generate a list of operating systems corresponding to said representative test failure and said at least one related test failure; and

instructions for linking said list of operating systems to said representative test failure in said database.

8. A method for classifying test results, comprising:

extracting data from a test result file, wherein said test result file identifies a failed attempt by a software application to conduct an electronic operation on a computer equipped with an operating system;

comparing said data to failure characteristics stored in a database; and

linking an operating system identification from said test result file to said failure characteristics if said data from a test result file matches said failure characteristics.

9. A method according to claim 8, further comprising marking properties of test result files to be ignored during said comparing.

10. A method according to claim 8, wherein at least one of said failure characteristics is an abstract characteristic that can be matched by a variety to data from a test result file.

11. A method according to claim 8, further comprising adding new failure characteristics to said database if said data from a test result file does not match said failure characteristics, wherein said new failure characteristics correspond to said data from a test result file.

12. A method according to claim 11, further comprising emphasizing said new failure characteristics in a GUI with respect to failure characteristics that are not new.

13. (Canceled)

14. A method according to claim 8, further comprising marking failure characteristics to indicate that a failure they represent is expected.

15.-21. (Canceled)

22. A method for classifying test results, comprising:

extracting data from a test result file, wherein said test result file identifies a failed attempt by a software application to conduct an electronic operation on a computer equipped with an operating system;

comparing said data from a test result file to failure characteristics stored in a database, wherein first data that identifies a test operation is used in said comparison and second data that identifies a test scenario comprising at least an operating system identifier is not used in said comparison; and

if a match is discovered from said comparing, identifying said data from a test result file and said failure characteristics as a single failure in a Graphical User Interface (GUI), and adding said operating system identifier to a list of operating system identifiers associated with said single failure.

23. A method according to claim 22, wherein said comparing is accomplished by a stored procedure in a database.

24. A method according to claim 22, further comprising cross-referencing said list of operating system identifiers such that it is accessible through said GUI from said single failure.

25.– 26. (Canceled)

27. A method according to claim 22, wherein said data from a test result file is in Extensible Markup Language ("XML") format.

EVIDENCE APPENDIX

The following evidence was relied upon by the Examiner in the final action dated April 19, 2007 to support the appealed rejections. Copies of the references are provided herewith.

U.S. Pub. 2002/0124213 (Ahrens)

U.S. Pat. 6,438,716 (Snover)

U.S. Pat. 7,058,860 (Miller)